UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/384,963	08/27/1999	SRIRAM SANKAR		8537

30505 7590 11/05/2003

MARK J. SPOLYAR
38 FOUNTAIN ST.
SAN FRANCISCO, CA 94114

EXAMINER	
ZHEN, LI B	

ART UNIT	PAPER NUMBER
2126	6

DATE MAILED: 11/05/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/384,963	Applicant(s) SANKAR ET AL.	
	Examiner Li B. Zhen	Art Unit 2126	

-- The MAILING DATE of this communication appears on the cover sh et with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on _____ .

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-40 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-40 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

2. Claims 1 – 5, 10 – 13, 16 – 20, 22, 24, 25 and 30 are rejected under 35 U.S.C. 102(a) as being anticipated by "TurboJ, a Java Bytecode-to-Native Compiler" (hereinafter Weiss).

As to claim 1, Weiss teaches [p. 114 Section 1; p. 115, 2nd paragraph; p. 116, 1st full paragraph] mixed-mode execution [mixed-mode execution of bytecode], comprising a higher-level mode of execution [TurboJ is an off-line Java-to-native compiler] and a lower-level mode of execution [Java Virtual Machine], in object-oriented programs [Java application], the higher-level mode of execution has components that are executed by the lower-level mode of execution [TurboJ relies on the JVM for...execution of any classes left in bytecode format] and it is possible to add new objects to a running program at the lower-level mode of execution [dynamic class-loading is permitted], the method comprising of: accessing/updating memory located in the lower-level mode of execution by code in the higher-level mode of execution [Turboized code requests services from the JVM via special entry points in the JVM; p. 116, 3rd full paragraph];

transferring control from the lower-level mode of execution to the higher-level mode of execution [JVM invokes Turbo-ized methods via the interludes; p. 116, 3rd full paragraph and Section 2.1]; and

transferring control from the higher-level mode of execution to the lower-level mode of execution [Interludes allow interpreted code to call Turbo-ized code...Turbo-ized code can also call interpreted code; p. 116, Section 2.1].

As to claims 10 and 24, Weiss teaches enabling mixed-mode execution of a computer program in an object-oriented programming environment [see the rejection to claim 1 above], a set of source code instructions [Java Source; p. 117] and a set of byte code instructions [interlude is generated as classfile, i.e. bytecodes; p. 117], comprising: a source code instruction processor (SCIP) to execute source code instructions [TurboJ is an off-line Java-to-native compiler];

a bridge class enabling addition of new classes to the SCIP [at run-time, dynamically loaded applets run in interpreted mode under the Turbo-ized HotJava; p. 118, 1st full paragraph];

a memory storing state information for the SCIP and the subject computer program [trampoline queries the JVM to find out the status of the callee; the JVM has this information by virtue of loading the interlude; p. 118, 1st paragraph];

a byte code instruction processor (BCIP) to execute byte code instructions [Java Virtual Machine] and retrieve, store, and/or modify data stored in the memory [Turboized code requests services from the JVM via special entry points in the JVM; p. 116, 3rd full paragraph];

the BCIP executes the SCIP and the subject computer program [TurboJ-generated code interfaces with a standard Java Virtual Machine; p. 114, Section 1, Introduction];

the SCIP identifies class members in the source code instructions of the subject computer program [TurboJ can determine the target of a call at compile-time; p. 119, 1st paragraph];

the SCIP is operable to extend the bridge class with methods allowing access to identified class members [Interludes allow interpreted code to call Turbo-ized code...Turbo-ized code can also call interpreted code; p. 116, Section 2.1]; and

the SCIP is operable to interpret a source code instruction [interludes allow interpreted code to call Turbo-ized code; p. 116, Section 2.1] and invoke methods in an instantiation of the bridge class to access required class or object attributes [JVM invokes Turbo-ized methods via the interludes; p. 116, 3rd full paragraph].

As to claim 2, Weiss teaches the new objects are specializations of existing objects with the ability to interact with the higher-level mode of execution through bridge methods in the lower-level mode of execution [if the original class file Foo.class declares a method bar(), then the corresponding interlude file is also named Foo.class and contains a method bar() with the native attribute; p. 116, Section 2.1].

As to claim 3, Weiss teaches the bridge methods have the ability to access or modify the memory [Turboized code requests services from the JVM via special entry points in the JVM; p. 116, 3rd full paragraph].

As to claim 4, Weiss teaches the bridge methods call methods in the higher-level mode of execution [JVM invokes Turbo-ized methods via the interludes; p. 116, 3rd full paragraph and Section 2.1].

As to claim 5, Weiss teaches the bridge methods call constructors in the higher-level mode of execution [class initialization causes a TurboJ initialization routine to be called; p. 116, Section 1.2].

As to claim 11, Weiss teaches the SCIP transferring execution of the subject computer program to the BCIP [Interludes allow interpreted code to call Turbo-ized code...Turbo-ized code can also call interpreted code; p. 116, Section 2.1], and the BCIP transferring execution of the subject computer program to the SCIP [JVM invokes Turbo-ized methods via the interludes; p. 116, 3rd full paragraph and Section 2.1].

As to claims 12, 16 and 25, Weiss teaches SCIP is operable to profile the source code and associate an index value to each class member [JVM spec for the getfield instruction says that the operands of the instruction are used to construct an index into the constant pool of the class file; p. 119, Section 3.2].

As to claims 13 and 17, Weiss teaches the SCIP is operable to extend the bridge class with methods calling to each class member [Interludes allow interpreted code to call Turbo-ized code...Turbo-ized code can also call interpreted code; p. 116, Section 2.1].

As to claim 18, Weiss teaches the byte code instruction processor is a virtual machine [JVM], and the memory is a component of the virtual machine [Java Virtual Machine, which provides memory and thread management; p. 114, Section 1].

As to claim 19, Weiss teaches the virtual machine is operative to execute byte code instructions to cause the memory management component, thread scheduling component or the thread synchronization component to perform tasks associated with data stored in the memory [TurboJ relies on the JVM for object management, thread management, class and library loading, and the execution of any classes left in bytecode format; p. 116, 1st full paragraph].

As to claim 20, Weiss teaches the SCIP transfers execution of the computer program to the BCIP by invoking a method in the bridge class, and the method invokes a corresponding method in the subject computer program [Interludes allow interpreted code to call Turbo-ized code...Turbo-ized code can also call interpreted code; p. 116, Section 2.1].

As to claim 22, Weiss teaches the BCIP executes a call to the SCIP and transfer execution of the computer program to the SCIP [JVM invokes Turbo-ized methods via the interludes; p. 116, 3rd full paragraph and Section 2.1].

As to claim 30, see the rejection to claims 10 and 24.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 6 – 9, 14, 15, 21, 23, 26 – 29, and 31 – 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Weiss in view of U.S. Patent No. 5,999,732 to Bak.

As to claim 32, Weiss teaches mixed-mode execution of a computer program by a SCIP and a BCIP, a bridge class allows for addition of classes to the SCIP, the bridge class includes a method calling to a first method in a subject computer program [see the rejections to claims 10 and 24 above]. Weiss does not teach determining at least one transfer point in the source code where control of execution may be transferred from the SCIP to the BCIP; inserting, during compilation of the source code, a control transfer method that executes the first method from the corresponding transfer point; and extending the bridge class with a method calling to the control transfer method.

However, Bak teaches determining at least one transfer point [mark the location] in the source code where control of execution may be transferred from the SCIP to the BCIP [system compiles the virtual machine instruction utilizing placeholder data in the place of the require runtime execution information...the system may also mark the location of the native machine instructions generated; col. 8, lines 22 – 30]; inserting, during compilation of the source code, a control transfer method [jump to a stub] that executes the first method from the corresponding transfer point [system overwrites the marked native machine instructions with a jump to a stub; col. 8, lines 43 – 54]; and extending the bridge class with a method calling to the control transfer method [system places native machine instructions in the stub to jump to the fixed native machine instructions; col. 8, lines 43 – 60].

It would have been obvious to a person of ordinarily skilled in the art at the time of the invention to apply the teaching of determining at least one transfer point in the source code, inserting a control transfer method, and extending the bridge class with a

method calling to the control transfer method as taught by Bak to the invention of Weiss because this will reduce the cost of dynamic class loading and initialization checks for compiled Java virtual machine instructions [col. 4, lines 15 – 20 of Bak].

As to claim 34, this is similar to claim 32 except control of execution is transferred from BCIP to SCIP [see the rejection to claim 32 above]. As to transferring control of execution from BCIP to SCIP, see the rejection to claim 1 above.

As to claim 37, this is a combination of method claims 24 and 34; see the rejections to claims 24 and 34 above.

As to claim 6, Weiss as modified teaches lower-level mode of execution invoke the bridge methods at predetermined locations [system compiles the virtual machine instruction utilizing placeholder data in the place of the require runtime execution information...the system may also mark the location of the native machine instructions generated; col. 8, lines 22 – 30 of Bak].

As to claim 7, Weiss as modified teaches control is transferred from the higher-level mode of execution to the lower-level mode of execution at method boundaries [mark the location of the native machine instructions generated at a step 511 so that after the method if compiled, the code for completing the native machine instructions may be generated; col. 8, lines 22 – 30 of Bak].

As to claim 8, Weiss as modified teaches control is transferred from the higher-level mode of execution to the lower-level mode of execution within the methods [if there are no more virtual machine instructions to compile, the method is compiled,

except for the incomplete native machine instructions marked; col. 8, lines 31 – 36 of Bak].

As to claim 9, Weiss as modified teaches transferring local state of the methods during the transfer of control from the higher-level mode of execution to the lower-level mode of execution within the methods [routine fixes the native machine instructions by replacing the placeholder data with the resolved class field information; col. 9, lines 48 – 55 of Bak].

As to claims 14, 15 and 29, Weiss as modified teaches the index value of each class member is associated with the corresponding method in the bridge class, and the bridge class are callable with reference to the associated index value [method for a class accesses entries in the constant pool by the index; col. 5, lines 33 – 41 of Bak].

As to claim 21, Weiss as modified teaches a control transfer method inserted within the computer program, the bridge class is extended to include a method calling to the control transfer method [system places native machine instructions in the stub to jump to the fixed native machine instructions; col. 8, lines 43 – 60 of Bak], and the SCIP transfers execution of the computer program to the BCIP upon execution of the control transfer method [Interludes allow interpreted code to call Turbo-ized code... Turbo-ized code can also call interpreted code; p. 116, Section 2.1 of Weiss].

As to claims 23, 35 and 38, Weiss as modified teaches the BCIP transfers the instruction associated with the call and any necessary parameters to the SCIP [routine fixes the native machine instructions by replacing the placeholder data with the resolved class field information; col. 9, lines 48 – 55 of Bak].

As to claims 26 and 28, Weiss as modified teaches each index value comprises a letter and a symbol [constant pool stores classes, methods, fields, and interfaces symbolically; col. 5, lines 42 – 50 of Bak].

As to claim 27, Weiss as modified teaches each index value comprises a number [each entry in the constant pool is indexed by a number starting with 1 and going up to the number of entries in the constant pool; col. 5, lines 33 – 42 of Bak].

As to claim 31, see the rejection to claim 10 and 24.

As to claims 33 and 40, see the rejection to claim 32.

As to claim 36, Weiss as modified teaches a process operative to determine whether control should be transferred [if there is a marker left, the system copies the marked native machine instructions to a buffer; col. 8, lines 37 – 43 of Bak].

As to claim 39, see the rejection to claim 32.

Conclusion

5. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

"Harissa: a Flexible and Efficient Java Environment Mixing Byctcode and Comipled Code" teaches mixing compiled and interpreted methods.

"The Java HotSpot Virtual Machine Architecture" teaches on-the-fly adaptive optimization technology.

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Li B. Zhen whose telephone number is (703) 305-3406. The examiner can normally be reached on Mon - Fri, 8am - 4:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John A. Follansbee can be reached on (703) 305-8498. The fax phone number for the organization where this application or proceeding is assigned is (703) 872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Li B. Zhen
Examiner
Art Unit 2126

lbz
October 29, 2003

JOHN FOLLANSBEE
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100